

ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



NGÀNH KHOA HỌC MÁY TÍNH

MÔN HỌC: NHẬP MÔN THỊ GIÁC MÁY TÍNH

HỌC KỲ I (2021-2022)

BÁO CÁO
PHÁT HIỆN NGƯỜI ĐI BỘ

Sinh viên 1:

Nguyễn Lâm Thảo Vy
MSSV: 19522547

Sinh Viên 2:

Nguyễn Thị Thúy An
MSSV: 19521183

Sinh Viên 3:

Hồ Mỹ Hạnh
MSSV: 19521470

Giảng viên:
Mai Tiến Dũng

Thành phố Hồ Chí Minh, tháng 1 năm 2021

Mục lục

1	Giới thiệu bài toán	2
2	Bộ dữ liệu	3
3	Các phương pháp	4
I	HOG	4
1	Giới thiệu về HOG	4
2	Pyramid - cách để detect người đi bộ ở cách khoảng cách khác nhau so với camera	5
3	Thực nghiệm	6
II	LBP	24
1	Giới thiệu về LBP	24
2	Các bước thực hiện	24
3	Thực nghiệm, đánh giá	25
4	So sánh	26
III	Faster R-CNN	29
1	Sơ lược về R-CNN và Fast R-CNN	29
2	Giới thiệu Faster R-CNN	31
3	Các thành phần của Faster R-CNN	32
4	Quá trình training Faster R-CNN	35
5	Thực nghiệm	35
4	Tài liệu tham khảo	39

Chương 1

Giới thiệu bài toán

Ngữ cảnh ứng dụng: Phát hiện người đi bộ là một ứng dụng cụ thể của phát hiện đối tượng. Trong những năm gần đây, với sự phát triển nhanh chóng của học sâu, công nghệ phát hiện người đi bộ cũng đã có những bước phát triển vượt bậc. Nó là nền tảng quan trọng trong nhiều lĩnh vực, chẳng hạn như lái xe không người lái, hệ thống giám sát thông minh, phân tích người đi bộ và phát triển robot. Đối với các hệ thống xe ô tô tự lái, việc phát hiện người đi bộ là rất quan trọng để xe có thể kịp thời né tránh để giảm thiểu tai nạn.

Input: Một bức ảnh chụp cảnh đường phố với góc nhìn từ camera hành trình xe hơi, trong ảnh có nhiều người và xe cộ qua lại trên đường phố.

Output: Cũng là bức ảnh đó nhưng có thêm các bounding box khoanh vùng vị trí từng người đi bộ có trong ảnh

Input



Output



Chương 2

Bộ dữ liệu

Data đã sử dụng: [Caltech Pedestrian Detection Benchmark](#)

Bao gồm các hình ảnh được cắt từ một video quay từ 30hz camera có kích thước khung hình là: 640x480px. Video được quay khi chạy qua giao thông thường ở đô thị.

Tất cả gồm 10 giờ quay video, hay 250 000 frame, với 350 000 bounding box và 2300 người đi bộ khác nhau.

Trong đó ta đã sử dụng:

- Sử dụng 3150 ảnh có người, người không quá nhỏ, không xuất hiện dày đặc.
- Sử dụng 3150 ảnh không người, được chọn ngẫu nhiên.

Chương 3

Các phương pháp

Trong bài báo cáo này, nhóm sẽ trình bày 3 phương pháp trích xuất đặc trưng cũng như detect ra bounding box chứa người đi bộ đã được ở trên. Ba phương pháp được sử dụng đó là:

- HOG
- LBP
- Faster R-CNN

I HOG

1 Giới thiệu về HOG

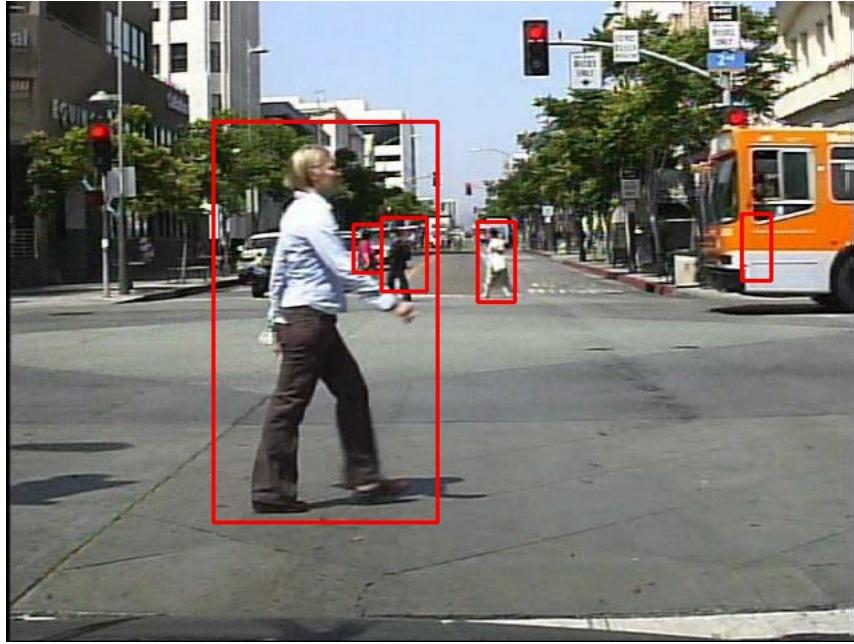
HOG là viết tắt của Histogram of Oriented Gradient - một loại “feature descriptor”. Mục đích của “feature descriptor” là trừu tượng hóa đối tượng bằng cách trích xuất ra những đặc trưng của đối tượng đó và bỏ đi những thông tin không hữu ích. Vì vậy, HOG được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của một đối tượng trong ảnh.

Bản chất của phương pháp HOG là sử dụng thông tin về sự phân bố của các cường độ gradient (intensity gradient) hoặc của hướng biên (edge directins) để mô tả các đối tượng cục bộ trong ảnh. Các toán tử HOG được cài đặt bằng cách chia nhỏ một bức ảnh thành các vùng con, được gọi là “tế bào” (cells) và với mỗi cell, ta sẽ tính toán một histogram về các hướng của gradients cho các điểm nằm trong cell. Ghép các histogram lại với nhau ta sẽ có một biểu diễn cho bức ảnh ban đầu. Để tăng cường hiệu năng nhận dạng, các histogram cục bộ có thể được chuẩn hóa về độ tương phản bằng cách tính một ngưỡng cường độ trong một vùng lớn hơn cell, gọi là các khối (blocks) và sử dụng giá trị ngưỡng đó để chuẩn hóa tất cả các cell trong khối. Kết quả sau bước chuẩn hóa sẽ là một vector đặc trưng có tính bất biến cao hơn đối với các thay đổi về điều kiện ánh sáng.

Gradient là 1 thông tin hữu dụng vì độ lớn (magnitude) của gradient ở các góc và cạnh luôn lớn hơn các vùng khác. Nói cách khác các góc và cạnh thường chứa nhiều thông tin hữu dụng cho object detection hơn các vùng khác trong ảnh.

2 Pyramid - cách để detect người đi bộ ở cách khoảng cách khác nhau so với camera

Ta sẽ dùng cửa sổ trượt để trượt qua ảnh trong bước detect và cửa sổ này là cố định. Nên nếu muốn detect người ở nhiều vị trí (xa, gần) khác nhau, ta cần trượt cửa sổ qua ảnh ở các tỷ lệ khác nhau.



Ta có thể thấy ảnh trên, người ở nhiều vị trí xa, gần khác nhau. Nhưng cửa sổ có kích thước cố định nên cách để detect được tất cả là điều chỉnh tỷ lệ của ảnh.

VD: Người phụ nữ gần với camera nhất trong ảnh có bounding box chắc chắn lớn hơn nhiều so với 64x128. Khi đó ta cần chỉnh ảnh về tỷ lệ nhỏ hơn. VD bounding box chứa người phụ nữ đó có kích thước 128x256 (tức gấp hai lần cửa sổ), ta sẽ giảm tỉ lệ thành 0.5 ảnh gốc để cửa sổ 64x128 trượt qua và lấy được khung hình chứa người phụ nữ đó. Tương tự với những bounding box nhỏ hơn cửa sổ thì ta sẽ lấy ảnh có tỷ lệ lớn hơn.

Việc lấy các tỷ lệ ảnh thích hợp giúp cho việc detect trở nên chính xác, hiệu quả hơn, ảnh hưởng rất lớn đến kết quả của detector. Tất nhiên người có thể xuất hiện trong ảnh với bất kỳ tỉ lệ nào, dường như không có giới hạn (giới hạn duy nhất ta có thể xác định là, tỉ lệ ảnh nhỏ nhất ta dùng được là kích thước ảnh bằng với cửa sổ, lớn nhất thì có thể giới hạn, ảnh càng lớn tức người đó càng xa camera, ta chỉ cần detect người trong một khoảng cách nhất định cần thiết). Tuy nhiên, số lượng tỷ lệ cần scale vẫn rất nhiều.

Thời gian và tài nguyên giới hạn nên ta chỉ chọn một số tỷ lệ đáng chú ý như trên để thực hiện.

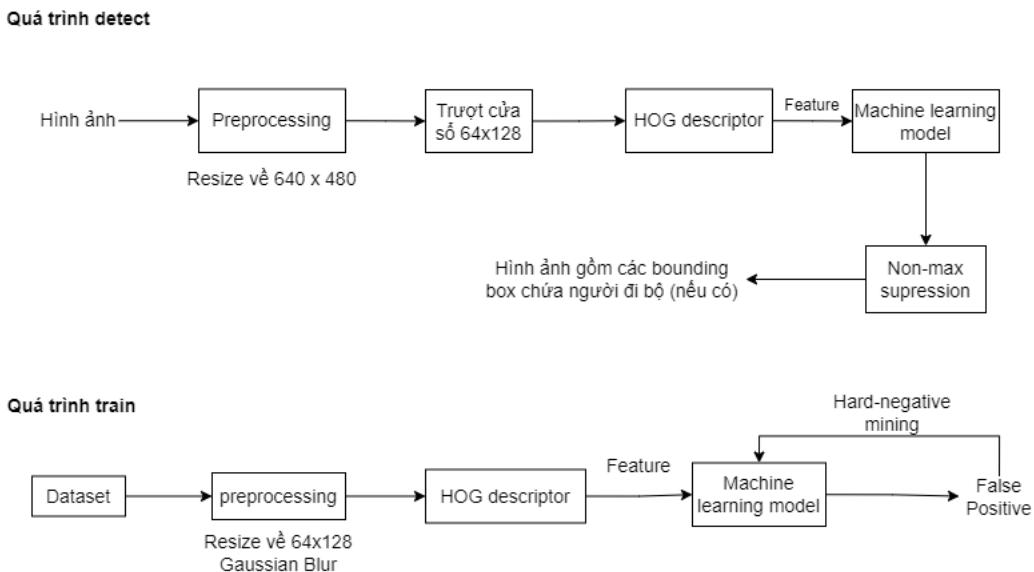
Tại sao các ảnh trên lại đáng chú ý? vì xe tự hành cần nhận ra người (trong các bounding box) đó để xử lý tình huống thích hợp: dừng xe, tránh kịp thời, phát ra kèn tín hiệu,...

Người ở khoảng cách xa cũng đáng chú ý, khi đó ta có thể tạo một hệ thống cho xe tính toán trước các tình huống có thể xảy ra để dự liệu, thực hiện các biện pháp an toàn (chạy chậm lại,...).

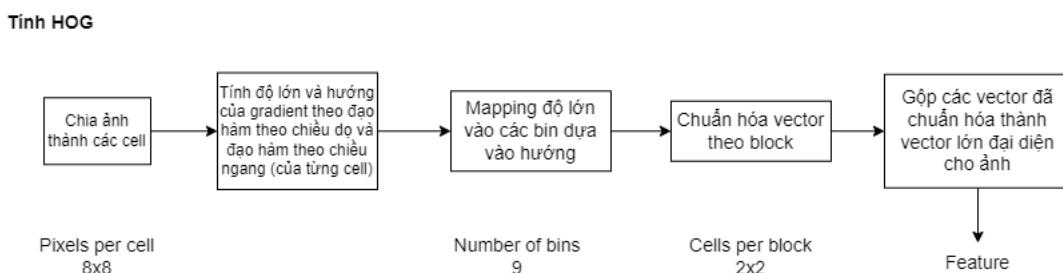
3 Thực nghiệm

Ở đây ta dùng HOG làm feature extractor và SVM để phân loại ảnh đó có phải là người đi bộ không. Phương pháp này ta dùng 1 cửa sổ kích thước cố định (64x128) trượt trên ảnh, rồi dùng HOG để trích xuất feature, và xem phần ảnh trong cửa sổ phải là người đi bộ không bằng cách đưa feature đã trích xuất ra được vào 1 classifier - ở đây là SVM.

Cả quá trình detect ảnh và quá trình train mô hình machine learning phục vụ cho việc dự đoán bounding box có phải chứa người hay không được tóm tắt sau đây:



Quá trình tính HOG:



3.1 Về cell size (pixels per cell) và mô hình machine learning

Về vấn đề này, ta thực nghiệm trên cùng bộ dữ liệu với nhiều cell size khác nhau và được kết quả sau đây:

So sánh accuracy score của các classifier trên các Pixels per cell khác nhau	SVM(Gaussian Kernel)	Linear SVM	Logistic Regression	Decision Tree
Classifier/Pixels per cell				
8x8	0.980578921	0.910935181	0.957694211	0.853661554
16x16	0.977733795	0.922563088	0.942107867	0.863433944
24x24	0.964374072	0.861083622	0.889905987	0.856135576
32x32	0.951756556	0.835230084	0.878154379	0.842528451

So sánh f1 score của các classifier trên các Pixels per cell khác nhau	SVM(Gaussian Kernel)	Linear SVM	Logistic Regression	Decision Tree
Classifier/Pixels per cell				
8x8	0.955385053	0.801434087	0.901213172	0.65917603
16x16	0.948453608	0.834741288	0.861538462	0.681109185
24x24	0.918042117	0.721270787	0.718532574	0.66416402
32x32	0.889580974	0.678881389	0.681332902	0.630907509

So sánh precision score của các classifier trên các Pixels per cell khác nhau		SVM(Gaussian Kernel)	Linear SVM	Logistic Regression	Decision Tree
8x8		0.968876081	0.788816504	0.929678188	0.678126852
16x16		0.969555035	0.788922156	0.912280702	0.702622169
24x24		0.932369942	0.647216036	0.824383164	0.684931507
32x32		0.898741419	0.595600677	0.805661821	0.653453453

So sánh recall score của các classifier trên các Pixels per cell khác nhau		SVM(Gaussian Kernel)	Linear SVM	Logistic Regression	Decision Tree
8x8		0.942264574	0.814461883	0.874439462	0.641255605
16x16		0.928251121	0.886210762	0.816143498	0.660874439
24x24		0.904147982	0.814461883	0.6367713	0.644618834
32x32		0.880605381	0.789237668	0.590246637	0.609865471

Ta có thể thấy cả hai kết quả của 2 vấn đề khác nhau:

- Accuracy, F1, Logistic Regression và Decision Tree đều có xu hướng giảm khi tăng số pixels per cell lên. Rõ ràng nhất là khi pixels per cell từ 8x8->32x32 thì tất cả các metric đều giảm mạnh.
- Về classifier ta cũng có thể thấy rõ SVM (Gaussian Kernel) có các metric như accuracy score, f1, precision và f1 đều cao hơn các classifier khác một cách rõ rệt.

3.2 Về số lượng bins trong chuẩn hóa theo block

cell 8x8, block 2x2, SVM (Gaussian kernel)								
Num_of_bins	Accuracy score	F1 score	Precision score	Recall score	Num_of_false_positive	Thời gian train	num_of_features	
3	0.955142785	0.902552841	0.884346423	0.921524664	215	3998.033	1260	
4	0.972074804	0.938594054	0.930578512	0.946748879	126	5620.843	1680	
16	0.967652262	0.92920354	0.917030568	0.941704036	152	15655.724	6720	
20	0.971822087	0.938346696	0.925804692	0.951233184	136	21527.824	8400	

Số lượng bins càng cao thời -> số feature càng lớn -> Thời gian train cũng như detect càng lâu

3.3 Non-max suppression

Đầu tiên là lọc theo confidence(phần trăm bounding box đó dự đoán là có người) theo một threshold nhất định. Ở đây ta dùng threshold=0.999 Chức năng dùng để loại bỏ các bounding box (bb) predict được cùng một đối tượng và chồng lấn lên nhau. Trong chức năng này ta kết hợp giữa IoU và confidence. Các bước thực hiện:

- Lấy ra một bounding box
- Tính IoU của bounding box đang xét với các bounding box còn lại
- Nếu IoU lớn hơn một threshold nhất định thì (ở đây ta dùng threshold=0.4): So sánh confidence . bounding box nào có confidence lớn hơn sẽ giữ nó lại để tiếp tục so sánh. Bounding box có confidence bé hơn sẽ bị loại bỏ.
- Cứ như thế tiếp tục lấy ra 1 bounding box trong các bounding box còn lại (có IoU với các bb trước đó nhỏ hơn threshold) và tiếp tục so sánh.

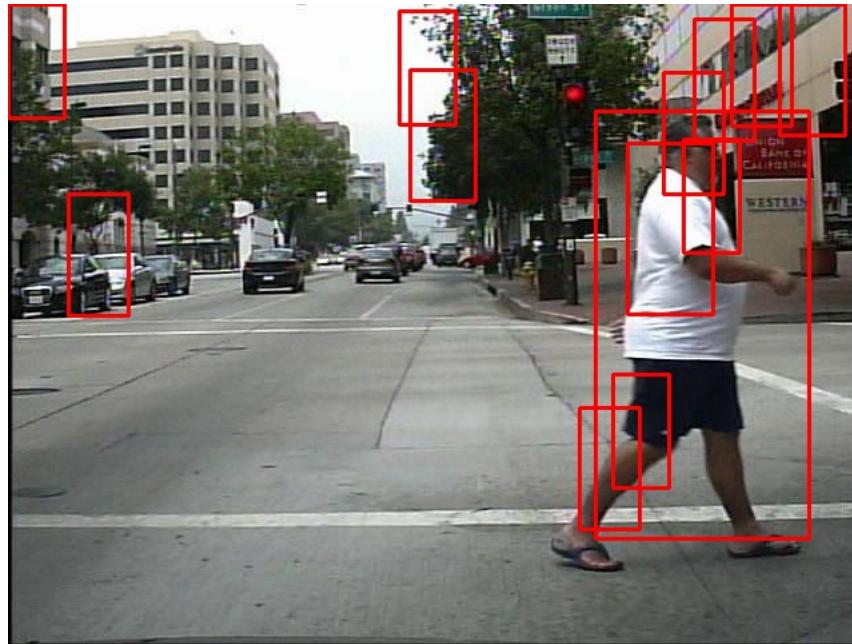
3.4 Hard-negative mining

Đây là quá trình training lại để giảm số lượng bb bị dự đoán sai (false positive). Bằng cách bỏ một số ảnh mới vào, nếu có 1 vector là false positive thì sẽ lưu lại để thực hiện quá trình train lại.

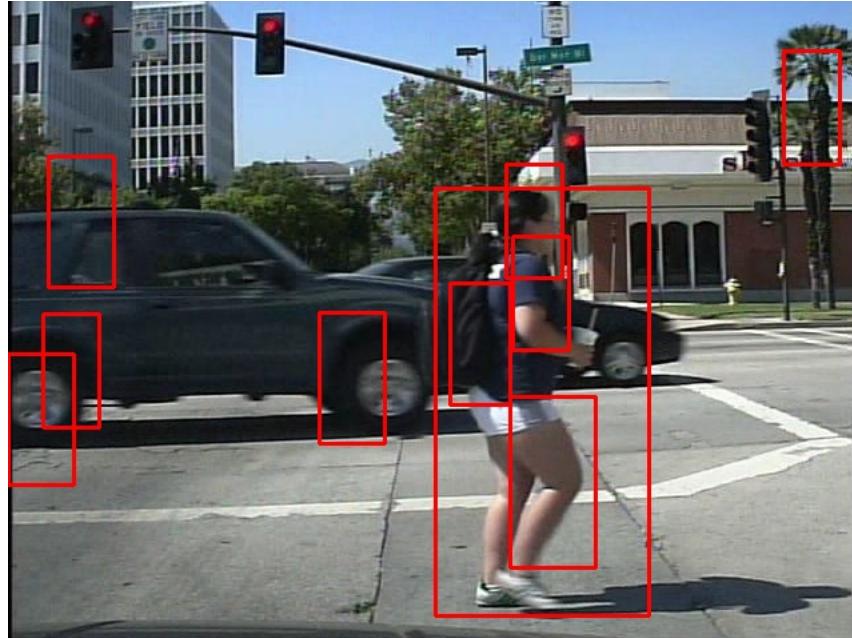
Ở đây ta chọn ra ngẫu nhiên 10 ảnh không có người để detect. Nếu có 1 vector detect là có người thì chắc chắn nó là false positive. Ta giữ nó lại để thực hiện retrain model. Sau đó suffer tất cả các vector thu được và chọn ra 500 ảnh để retrain.

3.5 Kết quả detect

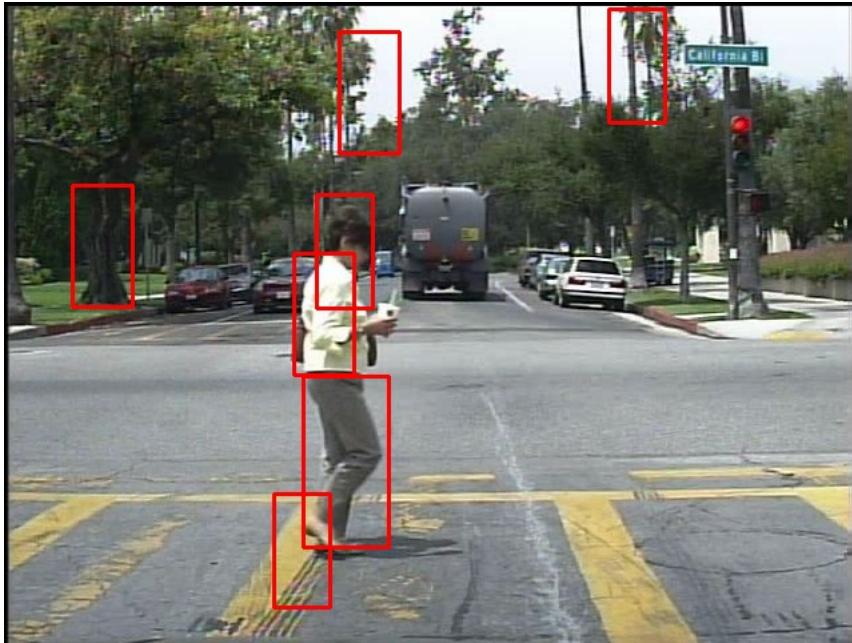
Trước khi hard-negative mining Kết quả detect sau khi áp dụng bộ descriptor HOG với pixels per cell=8x8, cells per block=2x2 và bins=9



Các bounding box có tập trung tại người đi bộ, tuy nhiên vẫn xuất hiện false positive.



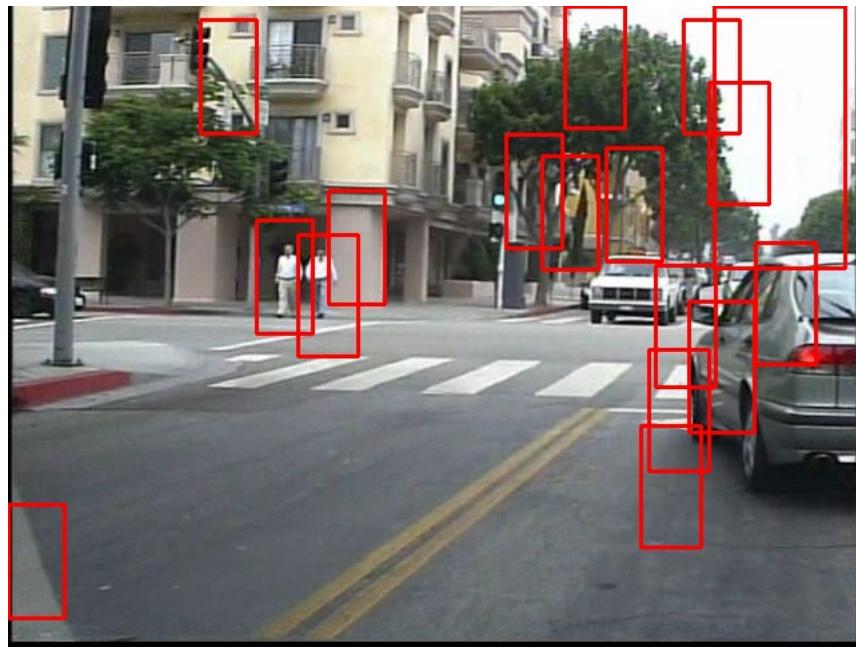
easy big 2 - Tương tự như ảnh easy big 1



easy big 3 - Tương tự như ảnh easy big 1 và 2. Tuy nhiên không có bounding box nào bao phủ cả người đi bộ như 2 trường hợp trên. Có thể là do không có tỉ lệ pyramid ảnh phù hợp.



easy small 1 - Có 1 người có bounding box là true positive, có 1 người không có bounding box (false negative). Tồn tại 1 số trường hợp false positive như các ảnh trước.



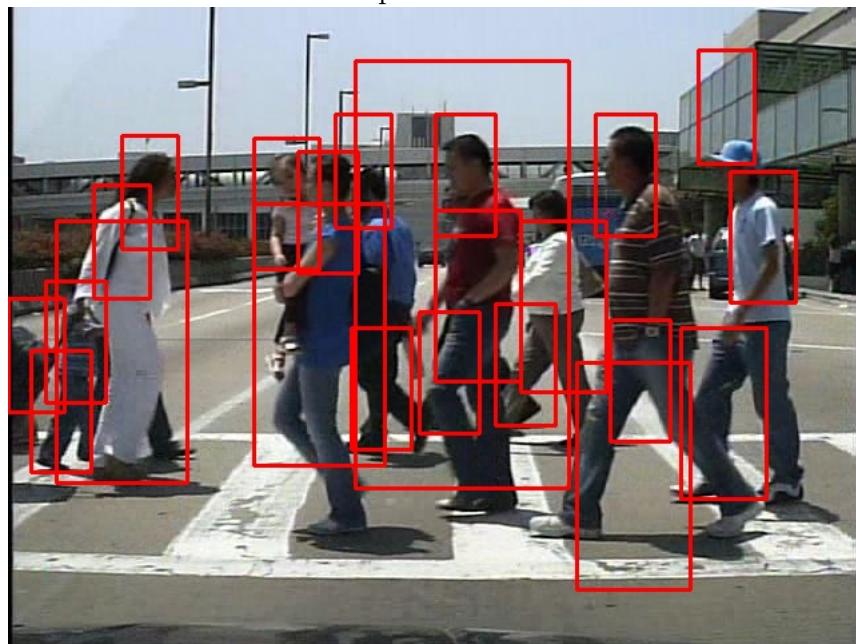
easy small 2 - 2 người trong ảnh có 2 bounding box là true positive. Tuy nhiên ảnh này tồn tại nhiều false positive hơn các ảnh còn lại



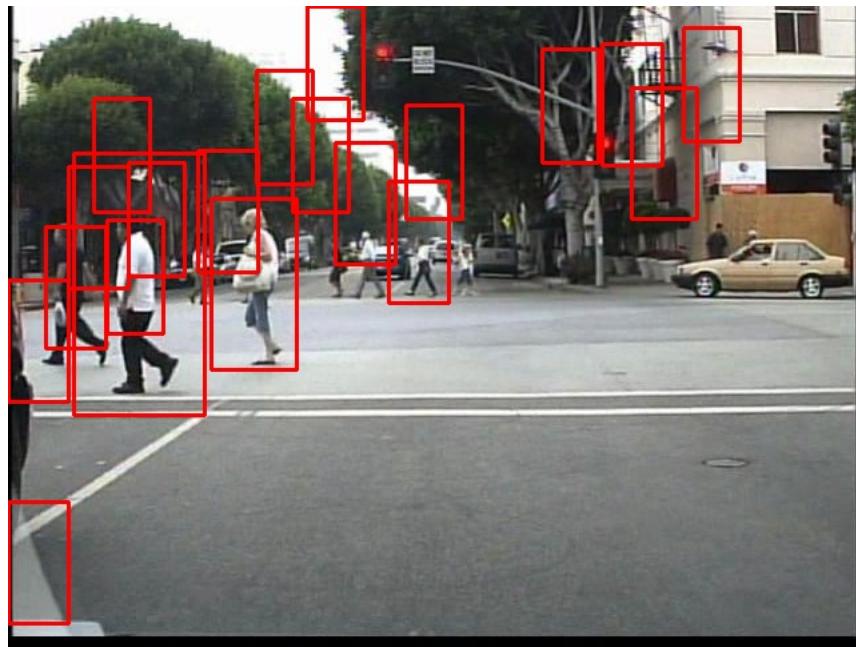
easy small 3 - Không thể đoán được cả 2 người trong ảnh, có 1 số false positive và 2 false negative



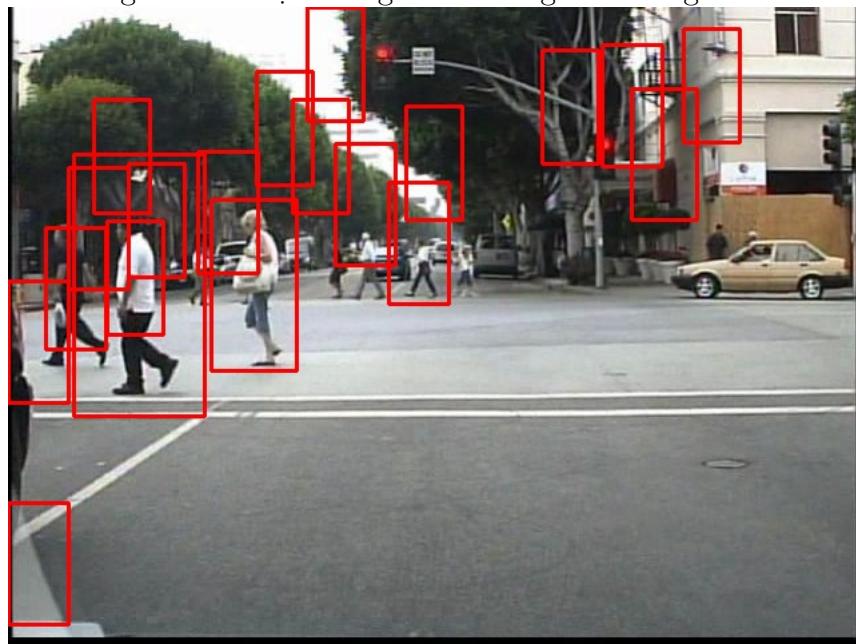
hard big 1 - Các bounding box tập trung vào người đi bộ, vẫn tồn tại false negative và false positive.



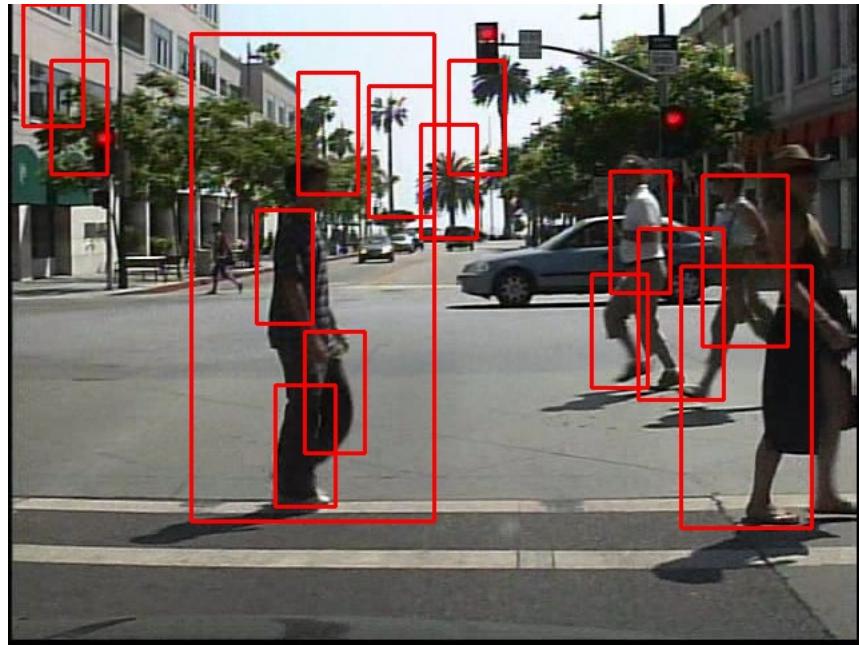
hard big 2 - Các bounding box tập trung vào người đi bộ, tuy nhiên vẫn chưa xác định trọn vẹn và chính xác.



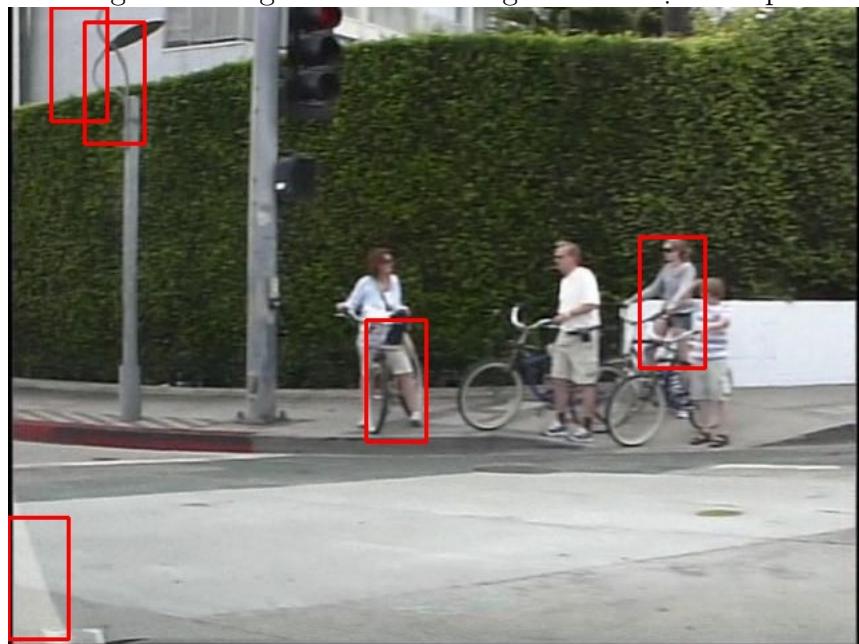
Có rất nhiều người đi bộ nhỏ ở khoảng cách khác nhau, tuy nhiên detector chỉ detect ra 2 người, những người còn lại ở quá xa/không có scale pyramid thích hợp và cả trường hợp detector không detect được dù người đó đứng kế bên người có thể detect.



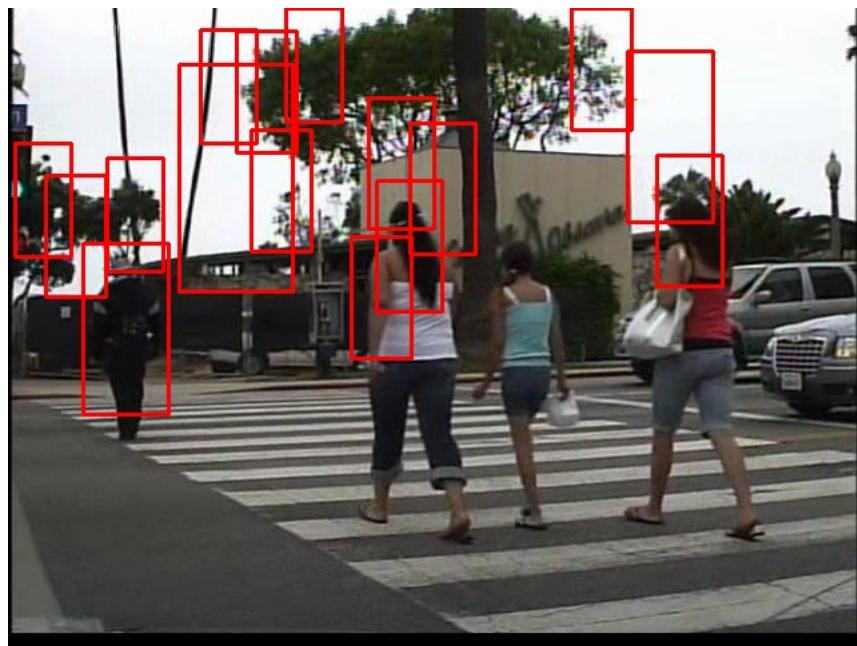
hard small 2 - chỉ detect được 1 số người, 1 số người không detect được. Có nhiều false positive



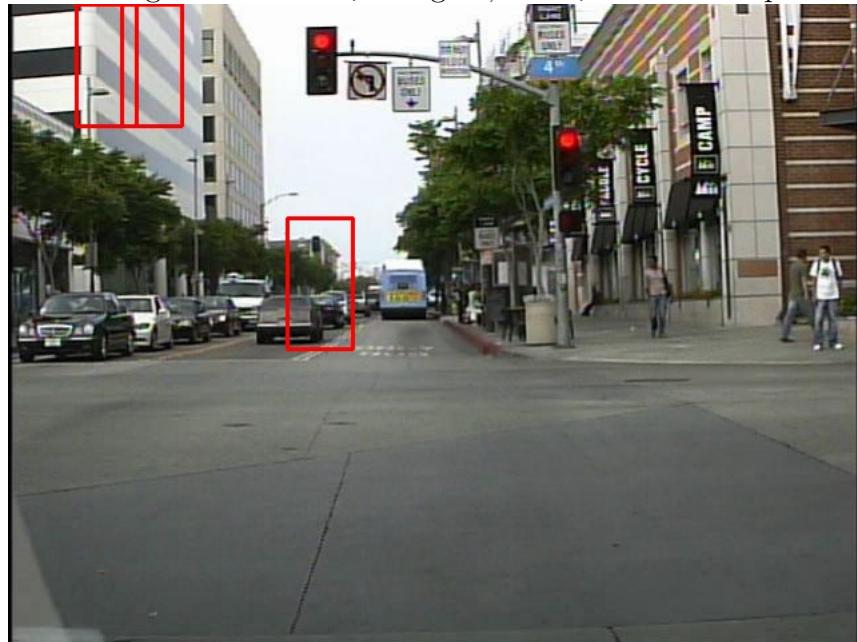
medium big 1 - Vẫn như các trường hợp ảnh có người ở khoảng cách gần khác, các bounding box tập trung vào người đi bộ, nhưng không detect chính xác, tồn tại nhiều bounding box chồng chéo nhau. Đồng thời tồn tại false positive



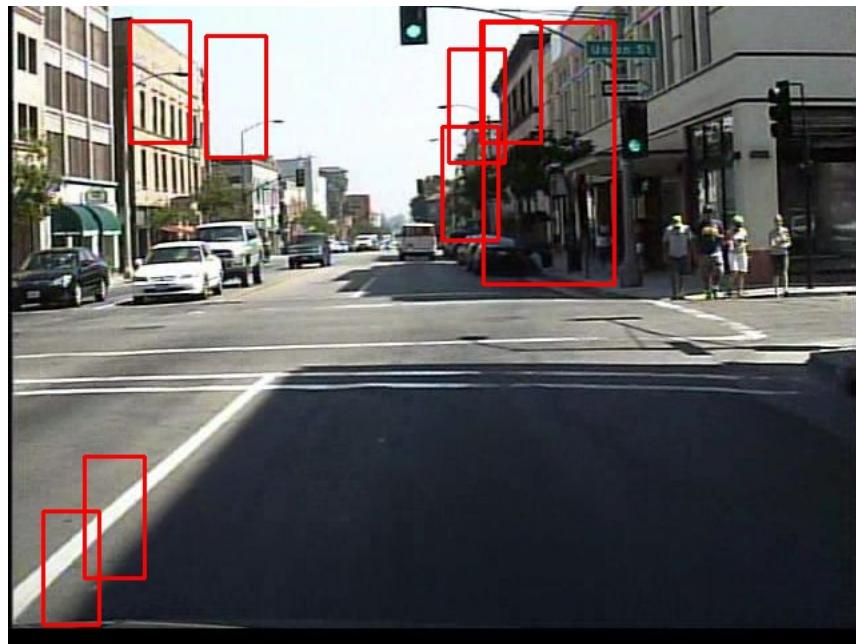
medium big 2 - Ảnh không gồm người đi bộ, chỉ có người đi xe đạp, tuy nhiên vẫn detect ra 1 số phần bộ phận là người đi bộ, những trường hợp này được tính là false positive



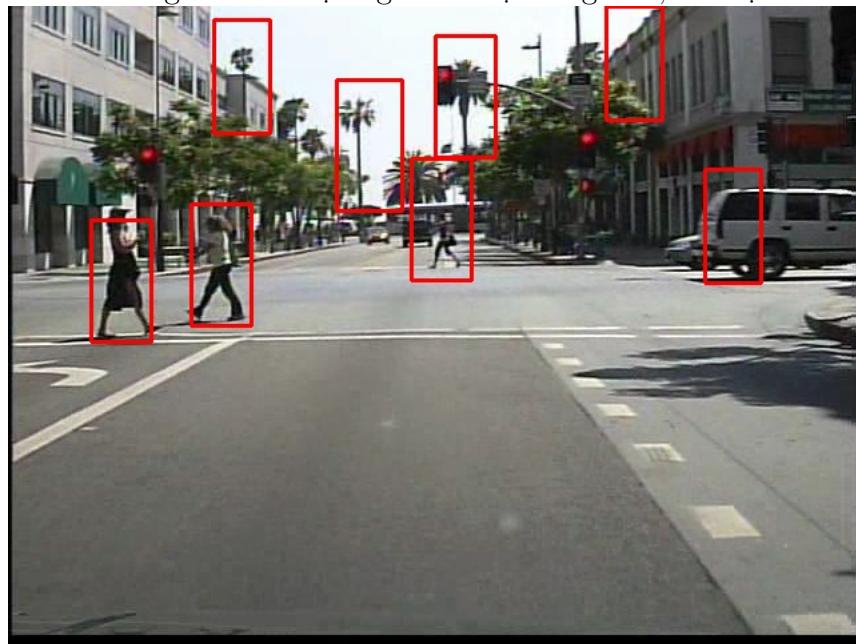
medium big 3 - detect được 1 người, tồn tại nhiều false positive



medium small 1 - không detect được người đi bộ trong ảnh, tồn tại 1 số false positive

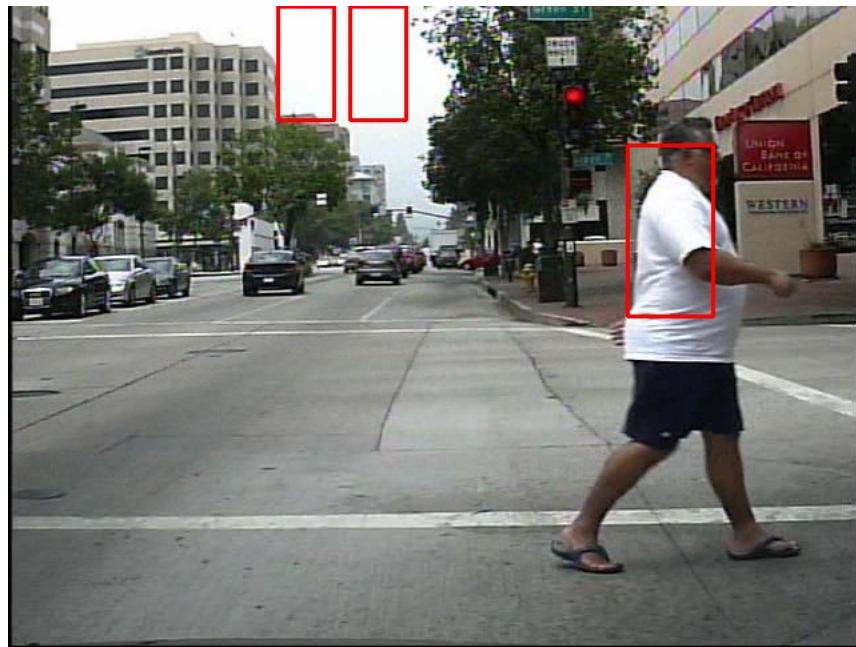


medium small 2 - không detect được người đi bộ trong ảnh, tồn tại 1 số false positive

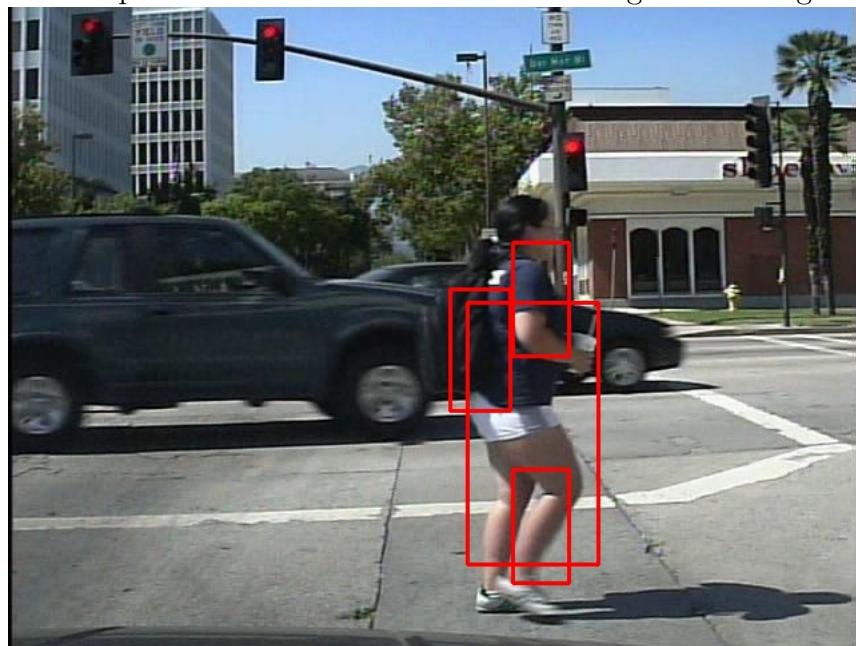


medium small 3 - Detect được tất cả người trong ảnh. Tuy nhiên vẫn tồn tại 1 số false positive

Sau khi hard-negative mining



easy big 1 - Có 1 bounding box tại chỗ người đi bộ, nhưng không bao được toàn bộ, tồn tại false positive như ít hơn trước khi hard-negative mining.



easy big 2 - không còn false positive như trước khi hard-negative mining, chỉ có các bounding box nhỏ tập trung tại người đi bộ, nhưng cũng mất đi bounding box lớn bao được toàn bộ người trong ảnh.



easy big 3 - Không còn false positive



easy small 1 - Chỉ detect được 1 người giống như trước khi hard-negative mining nhưng đã loại bỏ được tất cả false positive



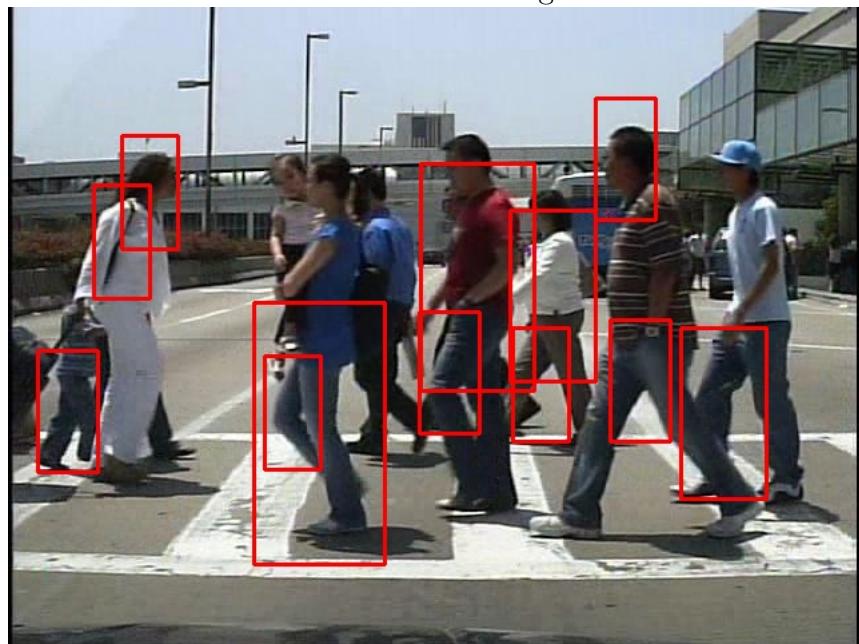
easy small 2 - Tất cả false positive đều được loại bỏ, nhưng 2 người trong ảnh lúc trước khi hard-negative mining detect được, thì bây giờ không detect được.



easy small 3 - Không thể đoán được cả 2 người trong ảnh, đã loại bỏ tất cả false positive



hard big 1 - Tuy đã loại bỏ hết false positive, tuy nhiên số lượng người đi bộ detect được ít hơn trước khi false negative.



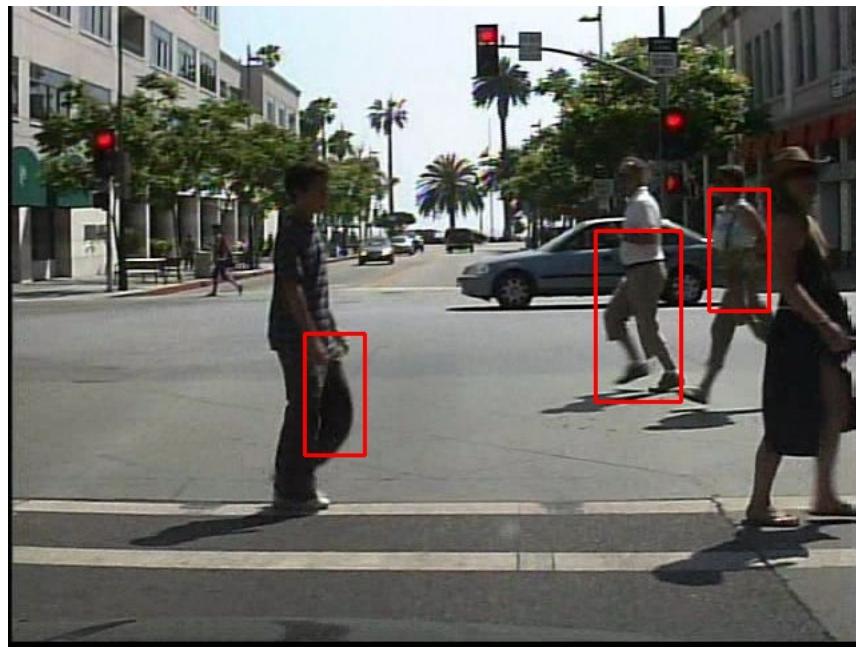
hard big 2 - chỉ có các bounding box nhỏ tập trung vào người đi bộ. Nhưng đã không còn false positive



hard small 1 - Có rất nhiều người đi bộ nhỏ ở khoảng cách khác nhau, tuy nhiên detector chỉ detect ra 1 người, những người còn lại ở quá xa/không có scale pyramid thích hợp và cả trường hợp detector không detect được dù người đó đứng kế bên người có thể detect. Vẫn còn FP dù đã qua hard-negative mining



hard small 2 - detect được 3/8 người trong hình. Không có false positive



medium big 1 - Đã loại bỏ false positive, tuy nhiên không detect ra người đàn ông ở gần nhất giống như model trước khi train hard-negative mining



medium big 2 - Đã loại bỏ hết false positive so với trước đó



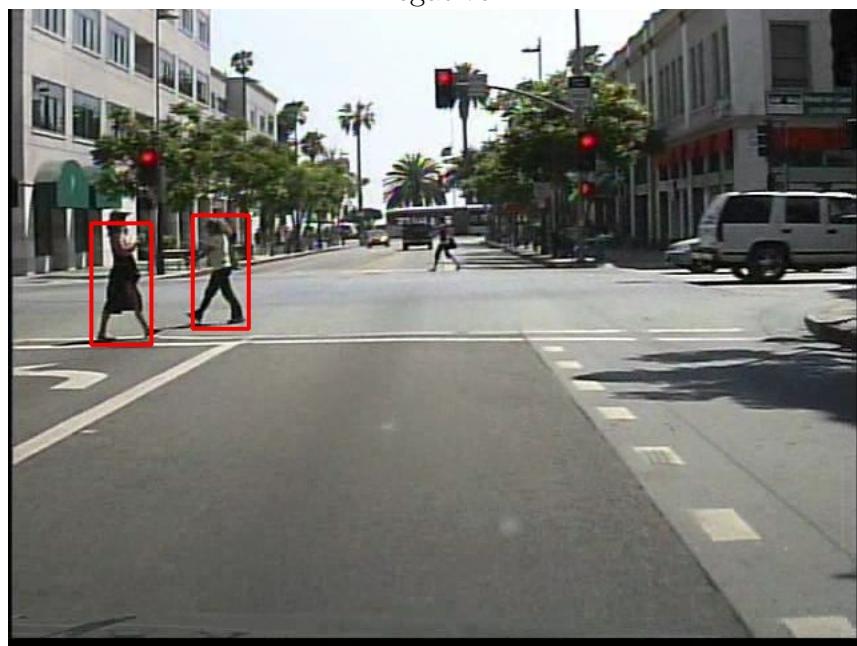
medium big 3 - Tuy false positive đã bị loại bỏ bớt, không có người nào detect ra được



medium small 1 - tương tự trước khi train hard-negative mining là không detect được người nào cả, nhưng đã loại bỏ false negative



medium small 2 - không detect được người đi bộ trong ảnh, nhưng đã loại bỏ false negative



medium small 3 - Đã loại bỏ hết false positive, tuy nhiên chỉ detect được 2 người thay vì 3 người trước khi hard-negative mining

II LBP

1 Giới thiệu về LBP

Local Binary Patterns (hay còn viết tắt là LBP) là một phương pháp rút trích đặc trưng trong xử lý ảnh, được Ojala trình bày vào năm 1996. Đặc trưng được rút trích sẽ tiếp tục được tiến hành chọn lọc (feature selection) thu gọn thành vector đặc trưng. Vector đặc trưng này sau đó có thể dùng để đưa vào mô hình học máy để học hay phân loại.

LBP là 1 toán tử cấu trúc đơn giản và hiệu quả gắn nhãn các pixel của ảnh bằng cách đánh giá các vùng lân cận của mỗi pixel và xem xét kết quả là một số nhị phân. LBP được dùng để đo độ tương phản cục bộ ảnh.

2 Các bước thực hiện

- Bước 1:

- Để xây dựng một LBP descriptor cần chuyển đổi input image về grayscale. Với mỗi pixel trong đó, chọn ra vùng lân cận (neighborhood) bao quanh pixel trung tâm đó. Một giá trị LBP sẽ được tính toán cho pixel trung tâm này và lưu vào một ma trận 2D có kích thước giống với input image.
- Khi so sánh các pixel trong vùng lân cận, nếu các pixel này có giá trị lớn hơn hoặc bằng pixel trung tâm thì sẽ được đánh dấu là “1”, ngược lại sẽ được đánh dấu là “0”.

3	2	6
1	4	5
7	1	4

0	0	1
0		1
1	0	1

Ví dụ: trong ảnh ta xét pixel 4, các vùng lân cận của nó là: $3 - 2 - 6 - 5 - 4 - 1 - 7 - 1$. Ta sẽ đi tìm trọng số (gọi là X) của mỗi điểm ảnh để tính toán giá trị nhị phân cần thiết. Với các vùng lân cận (tạm gọi là A), ta so sánh mỗi pixel với pixel đang xét (tạm gọi là B) (ở đây là pixel mang label = 4), nếu: $A \geq B \Rightarrow X = 1$ ngược lại nếu $A < B \Rightarrow X = 0$.

- Bước 2: Tính toán giá trị LBP cho pixel trung tâm

Sau khi chuyển đổi các pixel lân cận, ta có 1 chuỗi nhị phân 8 bit. Ta chuyển chuỗi nhị phân này về dạng thập phân ta được giá trị LBP. Với 8 pixel xung quanh pixel trung tâm, LBP sẽ có giá trị nằm trong khoảng [0, 255].

The diagram illustrates the conversion of a 3x3 LBP mask into a binary string and then into a decimal value. On the left, a 3x3 grid of binary values is shown:

0	0	1
6	7	0
0	5	1

An arrow points to the right, where the binary digits are listed vertically below powers of 2:

0	0	0	1	0	1	1	1	0
7	6	5	4	3	2	1	0	
2^4	2^3	2^2	2^1	2^0				

Below the binary digits is the decimal sum:

$$16 + 4 + 2 + 1 = 23$$

Sau khi có ma trận trọng số, ta sẽ có dãy nhị phân: 00010111 => số nhị phân: 23.

- Bước 3: Tính histogram

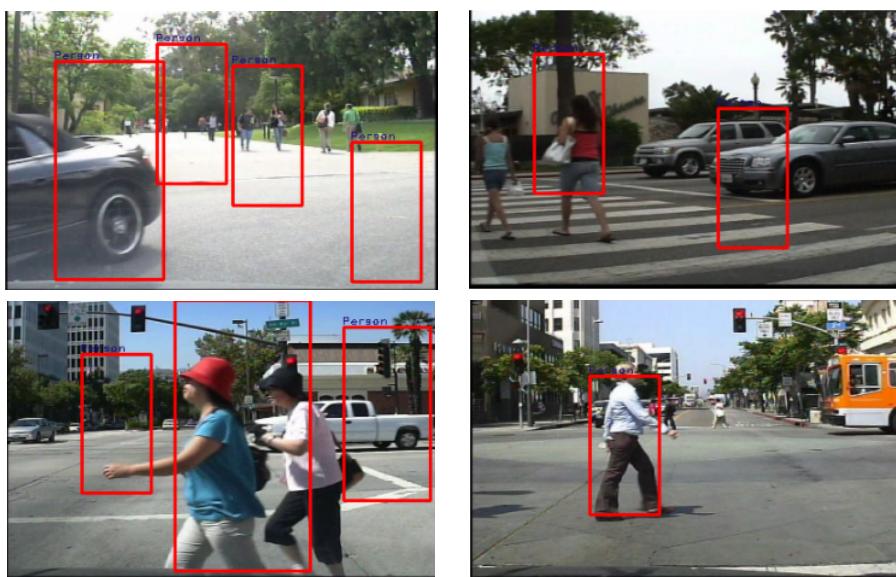
Tiếp theo chúng ta tính toán histogram cho ma trận 2D chứa các giá trị LBP của tất cả các pixel có trong input image. Kết quả nhận được sẽ là một Feature Vector có 256 chiều với mỗi vị trí trong vector tương ứng với số lần xuất hiện của nó trong ảnh.

3 Thực nghiệm, đánh giá

3.1 Thực nghiệm

Sử dụng đặc trưng LBP và SVM để detect ảnh. Nhóm đã thử qua nhiều phương pháp: Logistic Regression, Decision Tree, ... Và nhận thấy SVM cho ra kết quả tốt nhất và sử dụng nó để train bộ dữ liệu.

3.2 Kết quả

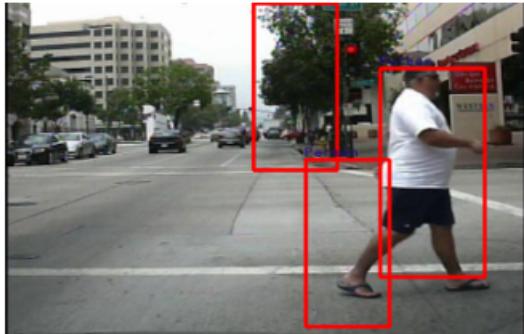


3.3 Đánh giá

- Ưu điểm: Thuật toán trích rút đặc trưng LBP cài đặt khá đơn giản, thời gian tính toán giá trị đặc trưng nhanh vì nó chỉ làm việc với giá trị nguyên.
- Nhược điểm: Độ chính xác không cao.
- Đề xuất: thêm đặc trưng HOG.

4 So sánh

Không chỉ sử dụng mỗi đặc trưng LBP, nhóm kết hợp thêm đặc trưng HOG và kiểm nghiệm xem đặc trưng mới có nâng cao chất lượng mô hình hay không. Dưới đây là một số hình minh họa để so sánh hai phương pháp:

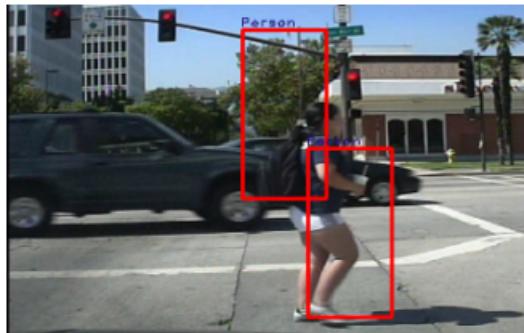


LBP



LBP + HOG

Như hình trên, với đặc trưng LBP ta thấy mô hình detect được đúng người tuy nhiên lại có thêm hai bounding box đã dự đoán sai. Sau khi được thêm đặc trưng số lượng bounding đã giảm nhưng vẫn thừa một cái, và bị chồng lấn với bounding box đúng.

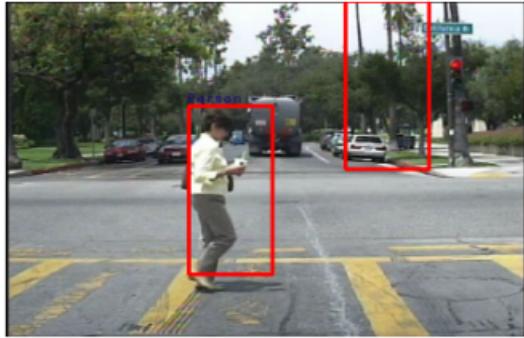


LBP



LBP + HOG

Qua tới ví dụ này, LBP+HOG đã làm rất tốt trong việc detect người chỉ trong một bounding box, tốt hơn so với chỉ có mỗi đặc trưng LBP.

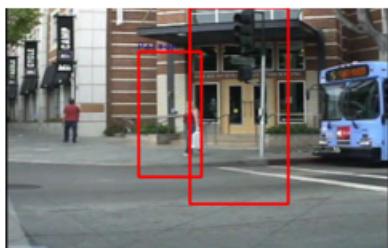
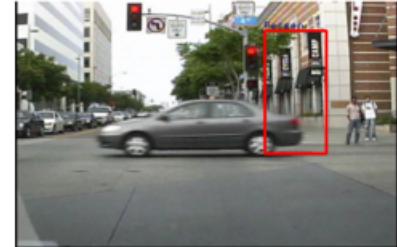
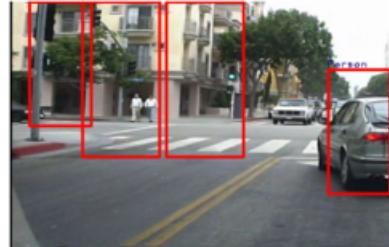
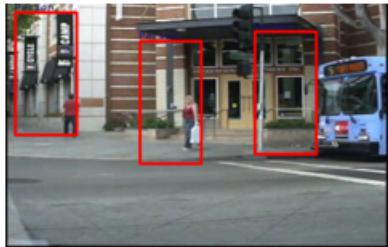


LBP

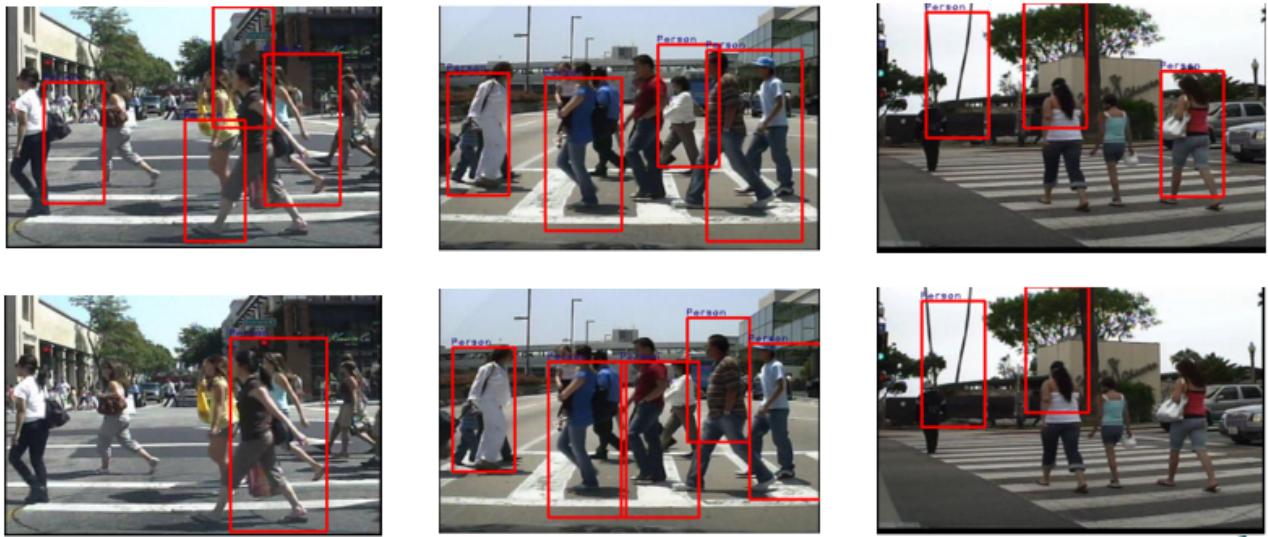


LBP + HOG

Tuy nhiên trong trường hợp này, khi mà đặc trưng LBP detect đúng người và dư một bouding box thì thay vì loại bỏ cái dư thừa, đặc trưng LBP+HOG đã loại bỏ tất cả các bouding box.



Hạn chế của mô hình ở đây là không xác định được kích thước người dẫn đến việc kích thước các bouding box quá khổ so với kích thước người cần detect.



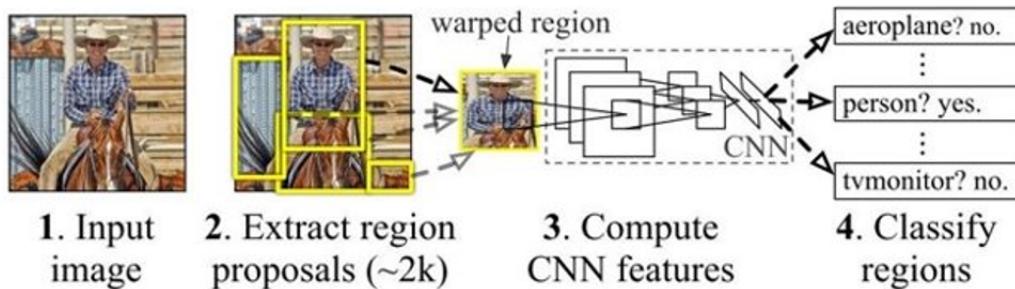
Một điểm hạn chế nữa chính là khó detect người khi mật độ xuất hiện người quá dày đặc. Dẫn đến việc chồng lấp các bounding box, detect trùng.

III Faster R-CNN

1 Sơ lược về R-CNN và Fast R-CNN

1.1 R-CNN

- R-CNN (Region-based Convolutional Neural Network) là một kiến trúc Object Detection được giới thiệu bởi Ross Girshick và các cộng sự vào năm 2013.
- Cách hoạt động của R-CNN:



Bước 1: Chuẩn bị ảnh input

Bước 2: Từ ảnh input, tạo các region proposals bằng thuật toán Selective Search.

Region proposals là các vùng đề xuất có thể chứa đối tượng thuộc một lớp nào đó, bao gồm cả background - lớp đối tượng đặc biệt trong các mạng R-CNN. Selective Search là một thuật toán tạo các region proposals bằng cách sử dụng kỹ thuật phân đoạn để nhóm các vùng trong ảnh có độ tương đồng về màu sắc, kích thước, hình dáng, mẫu mã,... Sau khi áp dụng Selective Search, ta có khoảng 2000 vùng đề xuất cho mỗi ảnh.

Bước 3: Dựa các region proposals này về kích thước cố định và sử dụng mạng CNN để trích xuất các features.

Mạng CNN này được sử dụng như một feature extractor dùng để trích xuất feature. Nó gồm nhiều convolutional layers, max pool layers, fully connected layers khác nhau tùy vào kiến trúc CNN mà ta sử dụng. Ví dụ một vài kiến trúc CNN: VGG16, VGG19, ResNet50, ResNet101,...

Bước 4: Sử dụng một mô hình phân loại để phân loại các features này thuộc vào background hoặc 1 trong các lớp đối tượng.

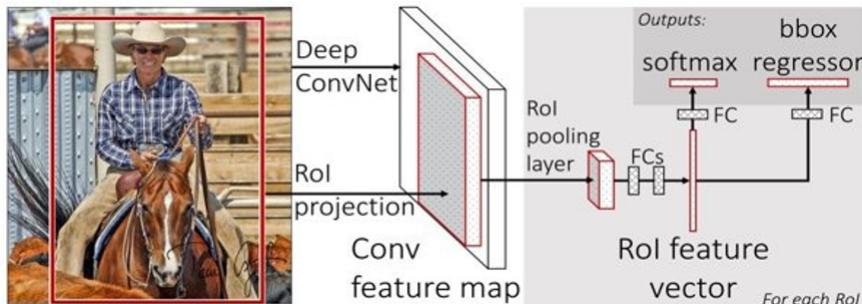
- Một vài khuyết điểm của R-CNN:

- Các bước hoạt động của R-CNN vẫn còn riêng lẻ và độc lập với nhau. Bước này xong rồi mới đến bước tiếp theo chứ chưa phải là một mô hình thống nhất hoàn chỉnh mà ta có thể huấn luyện từ đầu đến cuối.
- Cần dung lượng bộ nhớ lớn để lưu trữ các features trích xuất được từ 2000 region proposals cho mỗi ảnh.

Chính vì vậy, Fast R-CNN ra đời sau đó để khắc phục các nhược điểm này của R-CNN.

1.2 Fast R-CNN

- Fast R-CNN là một kiến trúc Object Detection được phát triển bởi Ross Girshick và được giới thiệu năm 2015.
- Cách hoạt động của Fast R-CNN:



Bước 1: Tạo các region proposals bằng thuật toán Selective Search

Bước này của Fast R-CNN giống với R-CNN. Nhưng từ các bước sau đó, bao gồm trích xuất feature và phân loại các features của các region proposals, thì Fast R-CNN sử dụng một network thống nhất như trong hình trên chứ không là còn là các bước riêng lẻ như R-CNN nữa.

Bước 2: Sử dụng một mạng CNN để trích xuất feature map từ các region proposals.

Bước 3: Feature map từ layer cuối của mạng CNN được đưa qua ROI pooling layer để trích xuất thành ROI feature vector có kích thước cố định

Các region proposals có các kích thước lớn bé khác nhau. Đối với R-CNN, ta phải đưa các proposals này về kích thước cố định trước khi trích xuất feature bằng CNN. Còn với Fast R-CNN, ta dùng CNN để trích xuất feature trước, sau đó đưa feature qua một layer đặc biệt gọi là ROI pooling layer để ra được feature vector có kích thước cố định.

Bước 4: Dưa feature vector đã trích xuất được qua 2 fully connected layers song song để ra 2 output vector cho mỗi region proposal:

- Softmax probabilities: xác suất phân loại cho mỗi class.
- Bounding box regression offsets: hiệu số hồi quy cho bounding box cũng chính là region proposal, gồm 4 giá trị: độ lệch tâm theo chiều x, độ lệch tâm theo chiều y, tỉ lệ lệch chiều rộng, tỉ lệ lệch chiều cao của ground truth box so với proposal tương ứng.

Output đầu tiên được lấy ra từ một fully connected layer sử dụng hàm kích hoạt là Softmax và có kích thước output là $k + 1$, tương ứng với $k + 1$ xác suất cho $k + 1$ class (bao gồm k class và 1 background).

Output thứ hai được lấy ra từ một fully connected layer sử dụng hàm kích hoạt là Linear và có kích thước output là $4*(k + 1)$, tương ứng với 4 hiệu số hồi quy cho $k + 1$ class. Ta dùng 4 hiệu số này để điều chỉnh tọa độ tâm và kích thước của region proposal sao cho nó về gần với ground truth box (bounding box gốc được label cho tập data).

- Khuyết điểm của Fast R-CNN:

Fast R-CNN vẫn còn hạn chế là sử dụng thuật toán Selective Search để tạo region proposals cho các ảnh input. Các proposals được tạo bằng thuật toán này thường có số lượng lớn nhưng hầu hết chỉ chứa background chứ không phải object nên gây tốn kém chi phí tính toán cho việc trích xuất và phân loại các features.

Chính vì vậy, Faster R-CNN ra đời nhằm khắc phục hạn chế này của Fast R-CNN.

2 Giới thiệu Faster R-CNN

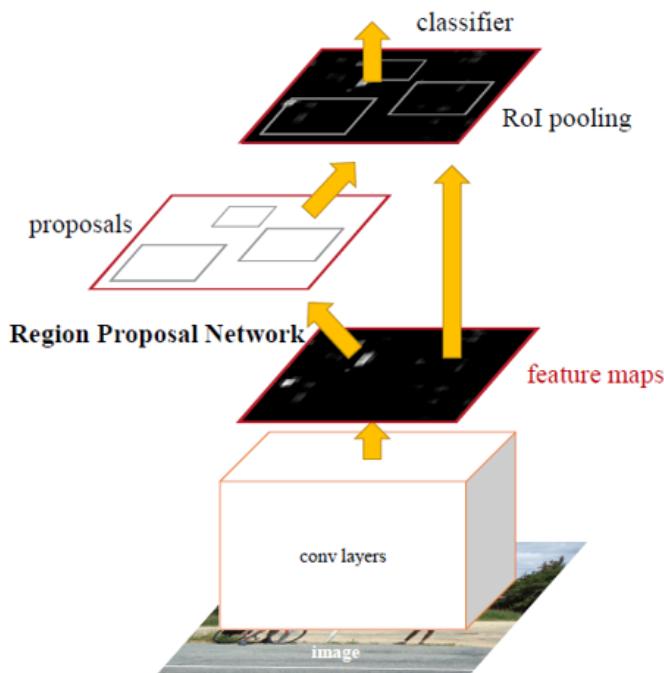
- Faster R-CNN là một kiến trúc Object Detection được giới thiệu bởi nhóm 4 tác giả Shaoqing Ren, Kaiming He, Ross Girshick và Jian Sun vào năm 2015.

Đây là phiên bản mở rộng của Fast R-CNN, cho tốc độ huấn luyện và detect object nhanh hơn Fast R-CNN nhờ vào sử dụng kiến trúc mạng đặc biệt gọi là Region Proposal Network (RPN) để tạo các region proposals thay cho thuật toán Selective Search.

- Kiến trúc Faster R-CNN gồm 2 phần:

- Region Proposal Network (RPN): tạo các region proposals là các vùng đề xuất có xác suất chứa object cao.
- Fast R-CNN: phân loại các region proposals được tạo bởi RPN xem chúng thuộc lớp đối tượng nào.

- Cách hoạt động của Faster R-CNN:



Bước 1: Dùng CNN để trích xuất feature map của ảnh input

Bước 2: RPN sẽ dựa trên feature map này để xuất ra các region proposals cho ảnh

Bước 3: Fast R-CNN lấy ra các feature maps của các proposals này và đưa qua ROI pooling layer để trích xuất thành các features có kích thước cố định.

Bước 4: Fast R-CNN đưa các features đã trích xuất qua một classifier có 2 fully connected layers song song ở cuối để ra 2 output vector cho mỗi proposal: xác suất class và 4 hiệu số hồi quy (như trong phần sơ lược về Fast R-CNN)

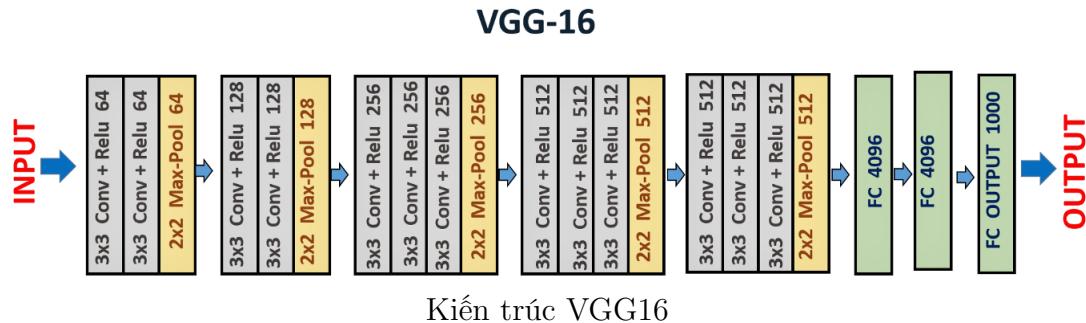
3 Các thành phần của Faster R-CNN

3.1 Region Proposal Network (RPN)

- Region Proposal Network (RPN) là một network dùng để tạo ra các region proposals. RPN có input và output như sau:

- Input: một ảnh có kích thước bất kỳ
- Output: một tập các region proposals là các bounding box đề xuất, bao gồm tọa độ các bounding box và xác suất có object trong mỗi box.
- Cách hoạt động của RPN:
 - Đầu tiên, RPN sử dụng một mạng CNN để trích xuất feature map từ ảnh input.

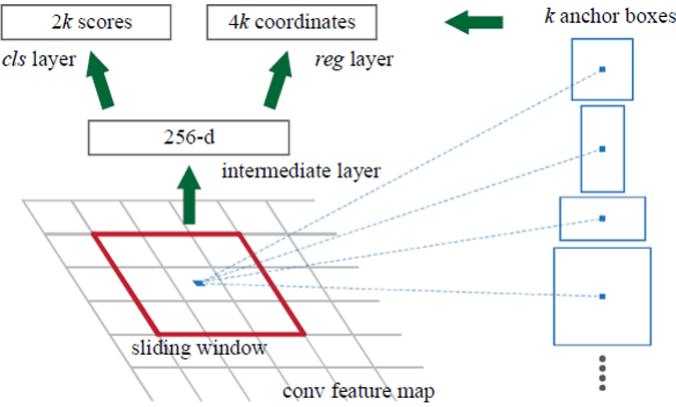
Mạng CNN được sử dụng như một feature extractor, dùng để trích xuất feature map từ ảnh input. Một trong những mạng CNN thường được sử dụng là VGG16, được giới thiệu bởi K. Simonyan và A. Zisserman đến từ Đại học Oxford vào năm 2014. Kiến trúc VGG16 gồm 5 khối convolutional layers và 3 fully connected layers theo sau. Các khối convolutional này sử dụng kernel kích thước 3×3 với số lượng filters tăng gấp đôi qua từng khối: 64, 128, 256 và 512 cho 2 khối cuối. Ở cuối mỗi khối convolutional là một max pool layer.



Kiến trúc VGG16

Thông thường, 3 fully connected layers cuối của mạng VGG16 được sử dụng cho mục đích phân loại ảnh nên không được dùng để trích xuất feature map từ ảnh input. Ta chỉ sử dụng 5 khối convolutional ở đầu mạng VGG16 và bỏ bớt đi max pool layer ở khối cuối cùng để sau khi mạng RPN tạo các proposals xong, ta sẽ thay vào đó là ROI pooling layer để Fast R-CNN trích xuất ra các features kích thước cố định.

- Theo sau mạng CNN là một convolutional layer có kích thước kernel 3×3 . Đây là một layer trung gian (intermediate layer) hoạt động như một sliding window duyệt qua toàn bộ feature map để chuẩn bị cho việc dự đoán các region proposals.



Sliding window duyệt qua feature map được trích xuất bởi mạng CNN

Ta có khái niệm anchor là vị trí chính giữa của sliding window này. Mỗi anchor sẽ ứng với các anchor box có tỉ lệ và kích thước khác nhau. Tỉ lệ của các anchor box có dạng là width : height và kích thước của chúng thường được biểu diễn bằng width (chia cho tỉ lệ để ra height tương ứng). Thông thường, các anchor box có 3 tỉ lệ là 1 : 1, 1 : 2, 2 : 1 và 3 kích thước là 128, 256, 512. Tại mỗi vị trí của sliding window trên feature map, ta sẽ dự đoán một số lượng tối đa k region proposals, cũng chính là k anchor box.

- Theo sau layer trung gian hoạt động như sliding window là 2 convolution layers song song có kích thước kernel 1x1 dùng để dự đoán các region proposals:
 - * cls layer: hoạt động như một binary classifier trả về một output vector có kích thước $2*k$, tương ứng với 2 giá trị 0, 1 cho mỗi anchor box. Đây là điểm dự đoán xem có object trong anchor box hay không.
 - * reg layer: hoạt động như một bounding box regressor, trả về một vector có kích thước $4*k$, tương ứng với 4 hiệu số hồi quy cho mỗi anchor box: độ lệch tâm theo chiều x, độ lệch tâm theo chiều y, tỉ lệ lệch chiều rộng, tỉ lệ lệch chiều cao giữa ground truth box so với anchor box tương ứng.

Sau khi lấy ra các anchor box có khả năng chứa object của cls layer, ta dùng 4 hiệu số hồi quy của reg layer để điều chỉnh vị trí và kích thước anchor box sao cho nó về gần nhất với ground truth box (các bounding box thật sự chứa object mà ta đã label cho tập data) để ra được các region proposals tốt nhất.

3.2 Fast R-CNN

- Sau khi mạng RPN tạo ra các region proposals, Fast R-CNN sẽ phân loại các proposals này xem chúng thuộc lớp đối tượng nào và điều chỉnh lại vị trí, kích thước của các proposals này để có được các bounding box dự đoán phù hợp nhất cho mỗi ảnh.
- Cách hoạt động của Fast R-CNN:
 - Đầu tiên, Fast R-CNN dùng một mạng CNN để trích xuất feature map từ ảnh input.

Mạng CNN này sẽ có cùng kiến trúc với mạng CNN của RPN, chẳng hạn nếu RPN sử dụng mạng CNN là VGG16 thì Fast R-CNN cũng dùng VGG16. Tương tự như RPN, mạng VGG16 trong Fast R-CNN chỉ sử dụng 5 khối convolution layers ở

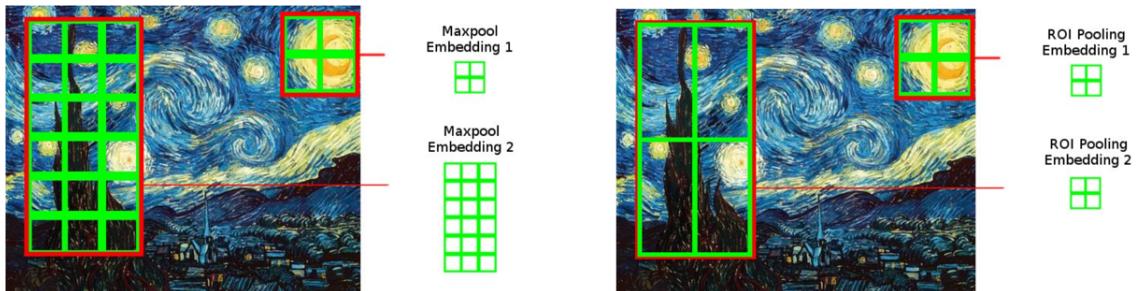
đầu chữ không dùng 3 fully connected layers phía sau. Tuy nhiên, RPN bỏ đi max pool layer cuối cùng ở khối convolutional thứ 5 còn Fast R-CNN thì thay max pool layer cuối của mạng CNN thành một layer đặc biệt gọi là ROI pooling layer để trích xuất ra các features kích thước cố định từ các region proposals được tạo bởi RPN.

- Sau đó, Fast R-CNN sử dụng các region proposals được tạo bởi RPN và lấy ra các feature maps của chúng rồi đưa qua ROI pooling layer để trích xuất thành các features có kích thước cố định.

ROI pooling layer là một dạng pooling layer cho ra output có kích thước cố định dù input có kích thước khác nhau.

Trong các kiến trúc CNN, max pool layer là loại pooling layer được sử dụng nhiều nhất. Với max pool layer, ta dùng một window có kích thước cố định trượt qua từng ROI (vùng quan tâm), tức là từng proposal được tạo bởi mạng RPN, và lấy ra giá trị max của window tại mỗi vị trí trượt. Vì các ROI có kích thước lớn bé khác nhau còn window có kích thước không đổi nên các features được trích xuất bởi max pool layer có kích thước khác nhau.

Còn với ROI pooling layer, mỗi ROI được chia thành các vùng nhỏ hơn sao cho số lượng vùng trong mỗi ROI bằng nhau trước rồi sau đó mới lấy ra giá trị max tại mỗi vùng. Nhờ vậy, dù các ROI có kích thước khác nhau nhưng các features trích xuất được vẫn có kích thước cố định không đổi.



Minh họa Max pooling (bên trái) và ROI pooling (bên phải)

- Theo sau ROI pooling layer là một vài fully connected layers và cuối cùng là 2 fully connected layers song song dùng để dự đoán các bounding box từ các region proposals được tạo bởi mạng RPN.

Hai fully connected layers song song này lần lượt cho ra 2 output vector cho mỗi bounding box dự đoán, cũng chính là mỗi proposal của mạng RPN:

- * Softmax probabilities: xác suất phân loại cho mỗi class.

Mỗi proposal được tạo bởi RPN có xác suất phân loại cao nhất cho class nào thì nó sẽ thuộc class đó.

- * Bounding box regression offsets: hiệu số hồi quy cho bounding box dự đoán, gồm 4 giá trị: độ lệch tâm theo chiều x, độ lệch tâm theo chiều y, tỉ lệ lệch chiều rộng và tỉ lệ lệch chiều cao của ground truth box so với bounding box dự đoán.

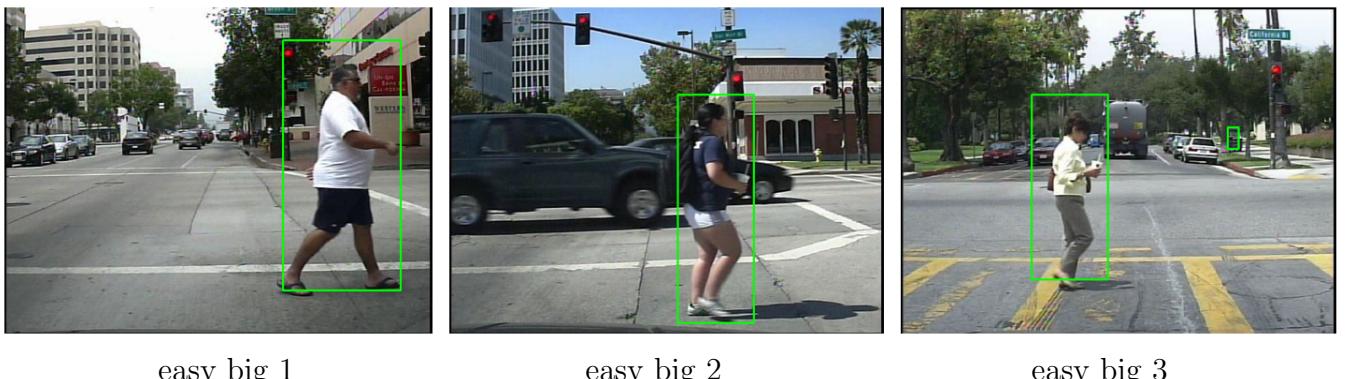
Dùng 4 hiệu số này để điều chỉnh tọa độ tâm và kích thước của các proposals được tạo bởi RPN về gần với ground truth box tương ứng của chúng để ra được các bounding box dự đoán sát với tập dữ liệu nhất.

4 Quá trình training Faster R-CNN

- Training data:
 - 2520 ảnh có người: được label sẵn khi tải tập dữ liệu Caltech Pedestrian Dataset
 - 2520 ảnh không có người: mỗi ảnh crop từ 1 - 5 bounding box tỉ lệ ngẫu nhiên (1:2, 1:2.5, 1:3) và chiều rộng ngẫu nhiên (16, 32, 46, 64, 96, 128, 192)
- Các siêu tham số đã thay đổi:
 - Tỉ lệ các anchor box: 1:2, 1:2.5, 1:3
 - Chiều rộng các anchor box: 16, 32, 46, 64, 96, 128, 192
 - Số region proposals cho mỗi ảnh: 20 (10 có người, 10 không người)
- Quá trình training:
 - Số epoch: 25 (2520 ảnh/epoch)
 - Thời gian train: 37.5 giờ (1.5 giờ/epoch)

5 Thực nghiệm

5.1 Kết quả



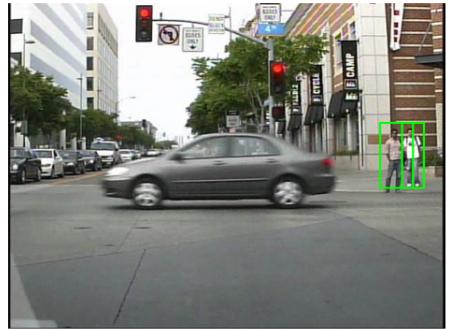
Với các ảnh dễ chỉ có 1 người duy nhất trong ảnh, Faster R-CNN cho kết quả detect khá tốt, bounding box bao quanh được toàn bộ người trong ảnh. Tuy nhiên đôi lúc mô hình vẫn còn detect sai một số chỗ, ví dụ như ảnh easy big 3 có một bounding box nhỏ không chứa người ở bên phải.



easy small 1

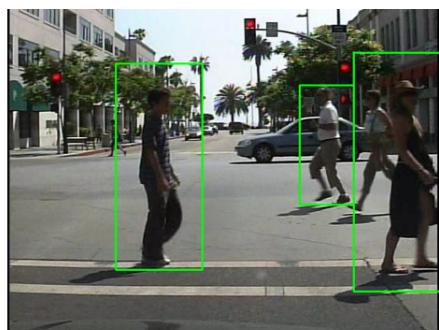


easy small 2

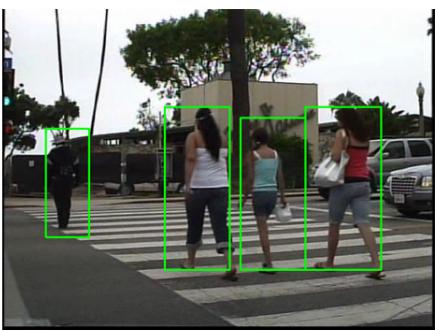


easy small 3

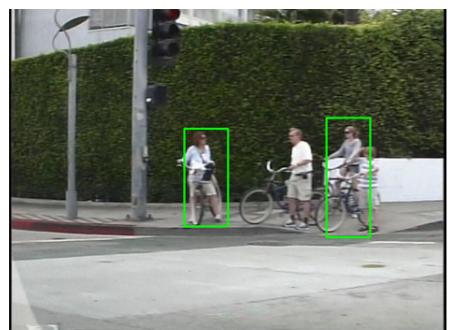
Với các ảnh dẽ chứa 2 người và kích thước người trong ảnh nhỏ hơn, Faster R-CNN vẫn cho kết quả detect khá ổn. Nhưng nếu kích thước người trong ảnh quá nhỏ, chẳng hạn như ảnh easy small 2, thì có thể sẽ không detect được người nào.



medium big 1

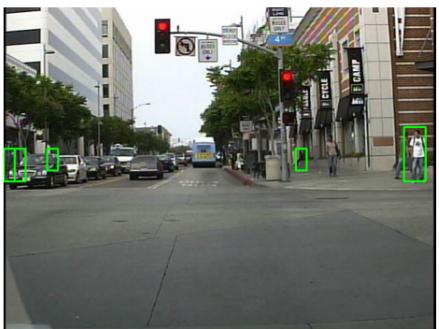


medium big 2

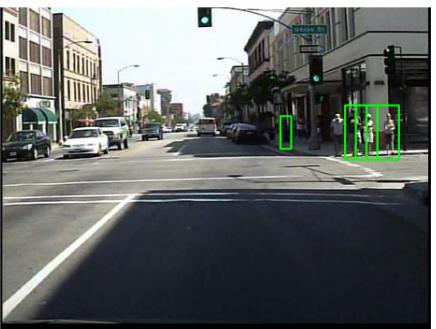


medium big 3

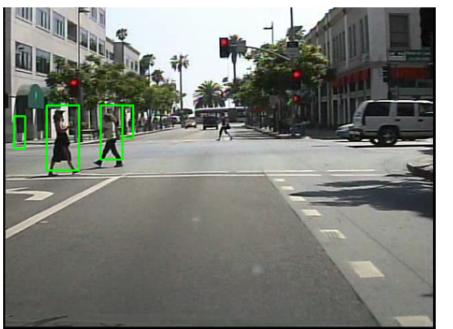
Với các ảnh trung bình có từ 3 - 5 người và kích thước người trong ảnh lớn, mô hình cho kết quả detect chưa được tốt lắm. Nếu vị trí những người trong ảnh cách nhau khá rõ ràng, như ảnh medium big 2, mô hình detect khá tốt và đầy đủ người. Nhưng nếu ảnh chứa những người đứng gần nhau như ảnh medium big 1 và medium big 2, thì mô hình có thể detect ra bounding box chứa cả 2 người trong đó (như medium big 1) hoặc chỉ detect được một trong 2 người do phải bỏ bớt đi bounding box còn lại (như medium big 3)



medium small 1



medium small 2



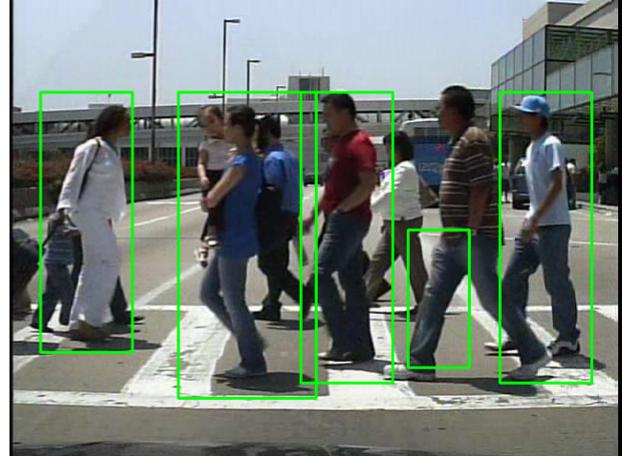
medium small 3

Với các ảnh trung bình có từ 3 - 5 người nhưng kích thước người nhỏ hơn, Faster R-CNN cho kết quả detect chưa được tốt. Cụ thể, mô hình không chỉ detect thiếu người, đặc biệt

là những người có kích thước quá bé, mà còn detect ra các bounding box không chứa người. Trong cả 3 ảnh medium small trên thì mô hình đều detect thiếu từ 1-2 người và detect sai một vài bounding box không chứa người.

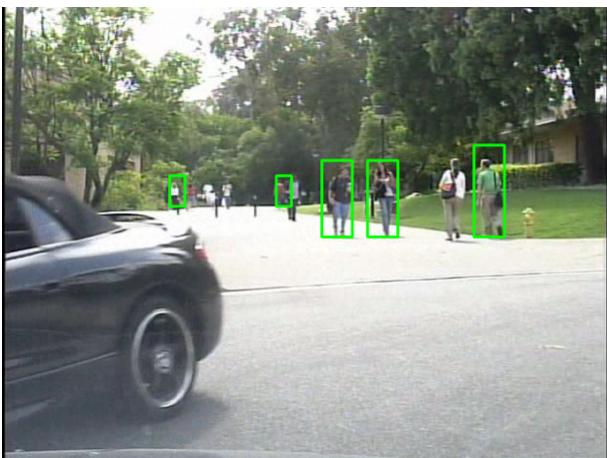


hard big 1

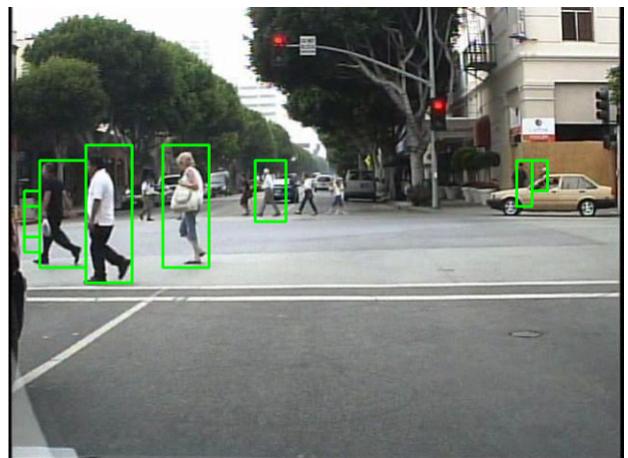


hard big 2

Với các ảnh khó chứa từ 5 người trở lên và kích thước người lớn, mô hình detect chưa được tốt lắm nếu như có nhiều người trùng lấp lên nhau, chẳng hạn như ảnh hard big 1 chỉ detect được 2 người đứng riêng lẻ ở bên trái. Còn nếu ảnh chứa nhiều người đứng tách biệt với nhau như ảnh hard big 2, mô hình detect tốt hơn hẳn mặc dù vẫn có bounding box bị detect sai hoặc chứa 2 người cùng lúc.



hard small 1



hard small 2

Với các ảnh khó chứa từ 5 người trở lên nhưng kích thước người nhỏ hơn, mô hình chỉ detect được những người ở phía trước. Còn những người có kích thước quá bé ở đằng xa, mô hình chỉ detect được từ 1-2 người và còn lại thì không detect được.

5.2 Nhận xét và đề xuất

- Nhận xét: Đối với tập dữ liệu Caltech Pedestrian Dataset, Faster R-CNN cho kết quả detect tương đối tốt với những ảnh có người không quá bé, không có người đứng quá gần hoặc trùng lấp lên nhau.

- Đề xuất: Để Faster R-CNN cho kết quả detect tốt hơn trên những ảnh như vậy, ta có thể:

- Bổ sung thêm vào training data những ảnh có người bé, ảnh có người đứng gần và trùng lặp lên nhau rồi train lại mô hình (trong quá trình chọn lọc data thì nhóm đã bỏ bớt ra những ảnh này).
- Resize ảnh input về kích thước lớn hơn, đồng thời tăng kích thước các anchor box để mạng RPN tạo ra các vùng đề xuất kích thước lớn hơn và mạng CNN của Fast R-CNN trích xuất được các feature maps chi tiết hơn từ các vùng đề xuất này.
- Sử dụng kiến trúc CNN phức tạp hơn để trích xuất được các feature map tốt hơn, chẳng hạn như kiến trúc VGG19, ResNet50, ResNet101,...
- Ngoài ra, có thể thay đổi kích thước output của ROI pooling layer trong mạng Fast R-CNN để ra được các feature có kích thước phù hợp hơn với tập data.

Chương 4

Tài liệu tham khảo

1. Histogram of Oriented Gradients and Object Detection - pyimagesearch
2. Bài 17 - Thuật toán HOG (Histogram of oriented gradient) - phamdinhkhanh
3. [Object Detection] 7 bước để xây dựng Single Object Detector với HOG và Linear SVM - ntppro's notes
4. Histogram of Oriented Gradients explained using OpenCV - LearnOpenCV
5. Feature Engineering for Images: A Valuable Introduction to the HOG Feature Descriptor - Analytics Vidhya
6. Hiện thực trích đặc trưng Local Binary Patterns (LBP)
7. Local Binary Patterns with Python OpenCV
8. Local Binary Patterns/github
9. Fast R-CNN
10. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
11. Faster R-CNN Explained for Object Detection Tasks
12. A deeper look at how Faster-RCNN works
13. Tìm hiểu về thuật toán R-CNN, Fast R-CNN, Faster R-CNN và Mask R-CNN